KANDIDAATINTYÖ

Koodinkattavuusanalyysi RTL-verifioinnissa

Joonatan Kangastie

Ohjaaja: Jukka Lahti

ELEKTRONIIKAN JA TIETOLIIKENNETEKNIIKAN TUTKINTO-OHJELMA
2020

Kangastie J. (2020) Koodinkattavuusanalyysi RTL-Verifioinnissa. Oulun yliopisto, Elektroniikan ja tietoliikennetekniikan tutkinto-ohjelma. Kandidaatintyö, 21 s.

TIIVISTELMÄ

Tässä tutkielmassa perehdytään siihen, mitä tarkoittaa koodinkattavuusanalyysi, minkälaisia eri tapoja on tutkia koodinkattavuutta ja miten niitä sitten hyödynnetään RTL-verifioinnissa. Lisäksi vielä tutkitaan erilaisten esimerkkien avulla Questa Sim ohjelman tarjoamia koodinkattavuusanalyysi menetelmiä.

Kangastie J. (2020) Code coverage analysis in RTL-verification. University of Oulu, Degree Programme in Electronics and Communications Engineering, Bachelor's Thesis, 21 p.

ABSTRACT

This thesis explains what code coverage is, goes through five different methods for analyzing code coverage and then gives examples how to use these methods in RTL-verification. Additionally there is simulation section in which Questa Sim simulator is used to demonstrate its tools for analyzing code coverage.

SISÄLLYS

ALKULAUSE

Haluaisin kiittää ohjaajaani Jukka Lahtea erinomaisesta ohjauksesta kandityössä. Lisäksi kiitän serkkuani Adam Kangastietä motivoinnista ja tuesta koko kandityöprosessin ajan.

Oulussa 20.3.2020

Joonatan Kangastie

.

LYHENTEIDEN JA MERKKIEN SELITYKSET

RTL	Register Transfer Level
SV	Systemverilog
FIR	Äärellinen impulssi vaste
IP-lohko	Valmiiksi suunniteltu ja toimivaksi todettu lohko

1. JOHDANTO

Digitaalipiirien monimutkaistuminen on pakottanut kehittämään toimivia ympäristöjä piirien varmentamiseen. Ilman oikeanlaista varmennusta suunnitellulle piirille ajautuu todennäköisesti isojakin virheitä, jotka voivat rikkoa koko piirin toiminnan. Varmentaminen on aloitettava jo suunnitteluvaiheessa, koska fyysiselle ja valmiille piirille muutosten tekeminen on vaikeaa. Tämä on johtanut tilanteeseen, missä digitaalipiirejä suunnittelevilla yrityksillä on pakottava tarve varmistaa piirien toimivuus. Käytännössä siis varmennuksen parissa työskentelee yleensä moninkertainen määrä ihmisiä verrattuna itse suunnittelutyön vaatimaan työntekijä määrään.

Varmennuksen helpottamiseksi on kehitelty erilaisia simulaattoreita, joista on tullut varmentamisen tärkeimpiä työkaluja. Simuloinnissa suunnitellulle digitaalipiirille tai lohkoille luodaan testiohjelma, jolla voidaan testata sen ominaisuuksia ja toimintaa. Oikeiden testitulosten saamisen lisäksi on tärkeää analysoida myös testien kattavuutta eli miten laajasti testiohjelma testaa suunnitelmaa.

Kattavuustyyppejä on olemassa kahdenlaista, funktionaalinen kattavuus ja koodinkattavuus. Funktionaalisessa kattavuudessa keskitytään siihen, miten hyvin testiohjelma on testannut suunnitelman eri ominaisuuksia ja toimintoja [2]. Tällä saadaan varmistettua suunnitelman toimivuus. Koodinkattavuudessa keskitytään pelkästään siihen, miten hyvin testiohjelma on käynyt digitaalipiiriä kuvaavaa koodia läpi.

Tässä tutkielmassa perehdytään tarkemmin toiseen näistä kattavuustyypeistä eli koodinkattavuuteen. Aluksi selitetään tarkemmin mitä on koodinkattavuus ja minkälaisia eri metodeja on analysoida koodinkattavuutta. Tämän jälkeen käydään vielä läpi Questa Sim -simulaattorin tarjoamat työkalut koodinkattavuuden analysoimiseen.

2. TEORIA

Koodinkattavuusanalyysimenetelmät ovat ensimmäisiä metodeja ohjelmistojen systemaattiseen testaamiseen. Kattavuusanalyyseillä saadaan tietoa siitä, miten hyvin koodia on käyty läpi simulointien aikana. Koodinkattavuudella ei kuitenkaan saada itse koodin toiminnasta paljoakaan tietoa eli se ei esimerkiksi kerro miten eri lohkot toimivat keskenään. Kattavuusanalyyseillä voidaan kuitenkin etsiä niin sanottuja reunatilanteita, joita normaaleilla testeillä ei saada testattua. [2]

2.1 Koodinkattavuusanalyysi menetelmät

Tässä osiossa käydään läpi suosituimpia koodinkattavuudenanalyysimenetelmiä ja niiden hyödyntämisen keinoja. Menetelmien nimet ovat lause-, haarautumis-, ehtolause-, tilanvaihto- ja tilakonekattavuus [4]. Jokaiselle menetelmälle on esitetty englanninkielinen vastine, koska suomalaisia nimityksiä ei käytetä juuri ollenkaan. Esimerkki koodeissa tarkasteltava rivi on merkattu punaisella.

2.1.1 Lausekattavuus

Lausekattavuus on yksinkertaisimpia kattavuuden mittareita. Sillä saadaan tietoa siitä, miten suuri osa koodin lauseista on suoritettu testien aikana. Lausekattavuutta voidaan verrata rivikattavuuteen, joka on vielä yksinkertaisempi, koska siinä käydään koodi läpi riveittäin. Lausekattavuuden ero tulee siitä, että lauseet voivat olla monta riviä pitkiä tai yhdellä rivillä voi olla monta lausetta. Tästä syystä lausekattavuutta pidetäänkin hyödyllisempänä kattavuuden mittarina. Käyttökohteena lausekattavuus luo hyvän yleiskuvan siitä, miten hyvin koodia on testattu, joten se kuluu kattavuusanalyysien perustyökaluihin. Jos kuvassa 1 olevaa esimerkki koodia testattaisiin ohjelmalla, joka ei anna ikinä A:lle arvoa 1, niin lausekattavuusanalyysi ohjelma ilmoittaisi, että punaista lausetta ei olisi käyty ollenkaan läpi. Englanninkielinen termi tälle on statement coverage. [1]

```
1   if (A == 1)
2     B = 1;
3   else
4     B = 2;
```

Kuva 1: esimerkki koodi lausekattavuudella

2.1.2 Haarautumiskattavuus

Haarautumiskattavuus tutkii koodista sen haarautumiskohtia ja mittaa ovatko Boolen muuttujat ohjausrakenteissa saaneet arvoiksi tosi tai epätosi. Ohjauslauseisiin kuuluu esimerkiksi SV kielessä if, case, while, repeat, forever, for ja loop -lauseet. Tyypillinen käyttökohde haarautumiskattavuudelle on esimerkiksi tarkistaa, että onko kaikki case-lauseen tapaukset suoritettu tai reset signaalin toimivuus. [1]

Kuvan 2 esimerkki koodissa keskitytään if-lauseeseen. Koodiin tulee kattavuusvajetta, jos testien aikana termi A ei saa arvoa 1, sillä se ei pääse niin sanotusti haarautumaan. Englannin kielessä haarautumiskattavuutta kutsutaan termeillä branch- tai decision coverage.

```
1  if (A == 1)
2      B = 1;
3  else
4      B =2
```

Kuva 2: Esimerkki koodi haarautumiskattavuudelle.

2.1.3 Ehtolausekattavuus

Ehtolausekattavuus kuvaa miten hyvin ehtolauseiden ehdot toteutuvat if-lauseissa. Ehtolausekattavuudella on olemassa kolme eri muunnelmaa. Yksinkertaisimmassa niistä kattavuusanalyysi keskittyy vain if-lauseessa tehtyyn päätökseen eli onko se saanut arvokseen toden ja epätoden. Tätä muunnelmaa voidaan pitää haarautumiskattavuuden jatkeena. Englanninkielinen termi tälle on condition coverage.

Hieman parempi metodi ehtolausekattavuus-analyysissä on keskittyä jokaiseen Boolen operaattoriin yksitellen. Kuvan 3 esimerkin if-lauseen tapauksessa se tarkoittaisi sitä, että kummatkin &-operaattorit käsiteltäisiin erikseen, jonka lisäksi vielä käsiteltäisiin ||-operaattori. Jotta tälle saataisiin 100% kattavuus niin jokaisen näistä pitää saada arvoiksi sekä tosi, että epätosi. Englanninkielinen termi tälle metodille on expression coverage.[4]

Paras ehtolausekattavuus menetelmä on kuitenkin niin sanottu FEC, joka tulee sanoista Focused Expression Coverage. Tässä metodissa jokaisen tulosignaalin kohdalla käydään läpi kolme asiaa. Minkälaisilla tulosignaalien kombinaatioilla tarkasteltava signaali voi vaikuttaa lähdön ulostuloon, jos se on asetettu arvoon 0. Sama käydään läpi tilanteessa, jossa signaali on asetettu arvoon 1. Sen jälkeen kerätään vielä talteen tieto siitä, että onko kaikki näistä kombinaatioista testattu. FEC on käytetyin ehtolausekattavuusanalyysimetodi, koska se käy ehtolauseet parhaiten läpi. Se on myös käytössä simulointiosiossa käytettävässä Questa Sim -simulaattorissa. [5]

```
if (A&B || C&D)
```

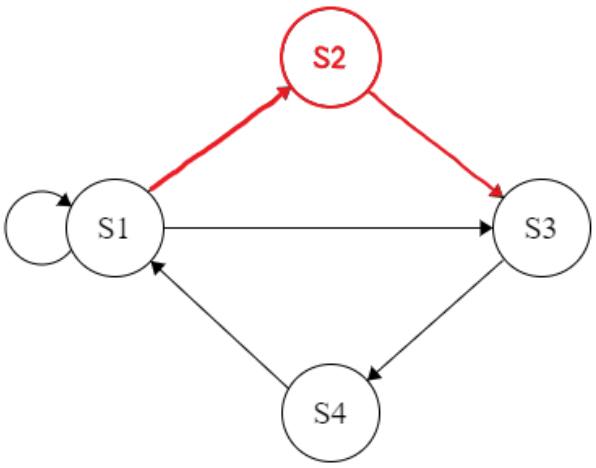
Kuva 3: Esimerkki if-lause

2.1.4 Tilanvaihtokattavuus

Tilanvaihtokattavuudella keskitytään logiikkabittien tilojen vaihdoksiin. Se kerää tietoa siitä, miten monesti jokin tietty bitti on vaihtanut tilaansa. Jos esimerkiksi jokin bitti on vaihtanut tilansa 0:sta 1:seen, mutta ei enää takaisin, niin kyseisen solmun kattavuus on 50%. Tällä voidaan myös tarkastella muutoksia korkean impedanssin tilaan eli Z:taan. Käytetään yleensä IP lohkojen välisten kytkentöjen testaamiseen [1] tai sitten tehonkulutuksen analysointiin [2]. Englanninkielinen termi tälle on toggle coverage.

2.1.5 Tilakonekattavuus

Tilakonekattavuus mittaa tilakoneessa tapahtuvia tilanvaihdoksia ja tiloja missä ohjelma on käynyt. Jotta saataisiin 100% kattavuus niin tilakoneessa pitää olla käytynä kaikki tilat ja mahdolliset tilojen vaihtumiset läpi. Tilakonekattavuus on hyvä tapa kartoittaa tilakoneiden toimivuus, sillä niissä voi olla vakavampiakin virheitä. Kuvan 4 esimerkki tilakoneessa on kattavuusvajetta, koska tilassa S2 ei olla käyty kertaakaan. Testeihin pitäisi koodata tilanne missä käytäisiin S2 tilassa edes kerran, jotta kattavuusvaje saataisiin korjattua. [1]



tila	käyntikerrat
S1	50
S2	0
S3	10
S4	10

Tilan vaihtuminen	kerrat
S1 -> S1	40
S1 -> S2	0
S1 -> S3	10
S2 -> S3	0
S3 -> S4	10
S4 -> S1	10

Kuva 4: Esimerkki tilakone

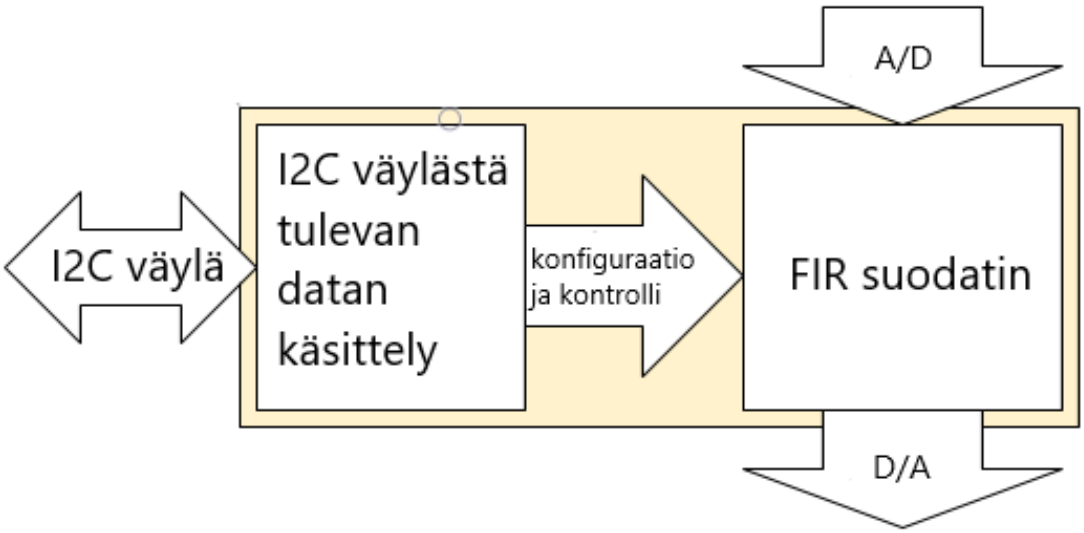
3. SIMULOINTI

Tässä osiossa perehdytään erilaisin esimerkein Questa Sim ohjelman koodinkattavuuden analyysimenetelmiin. Simuloitavana ohjelmana toimii SV:lla koodattu FIR-suodatin, jolle on tehty testipenkki sen testaamista varten.

3.1 Simuloitava ohjelma ja testipenkki

Kuvassa 5 on ohjelman rakenne yksinkertaistettuna. I2C-väylän avulla saadaan ennalta määritetyt suodattimen kertoimet, jotka sitten käsitellään. Datankäsittelylohkossa I2C-data synkronoidaan, josta sitten päätellään, lähetetäänkö I2C-väylältä käskyjä vai dataa. I2C:sta tullut data sitten tallennetaan rekistereihin, joita FIR-suodatinlohko käyttää hyväkseen. Suodatinlohkossa on laskemiseen tarvittavat lohkot ja kontrolliosa.

Testiohjelmassa testataan koko suodattimen toimintaa. Testin ideana on tallentaa suodattimen kertoimet rekistereihin muistiin. Tämän jälkeen suodattimen toimivuutta testataan erilaisilla signaaleilla, kuten askeleilla, impulsseilla ja taajuuspyyhkäisyllä



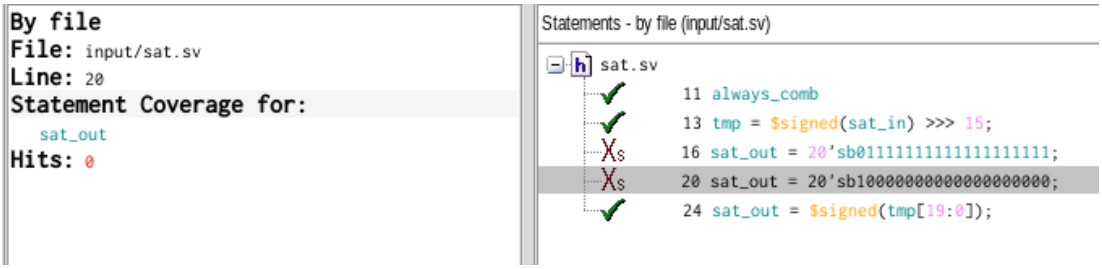
Kuva 5: Simuloitavan ohjelman yksinkertaistus

3.2 Esimerkit

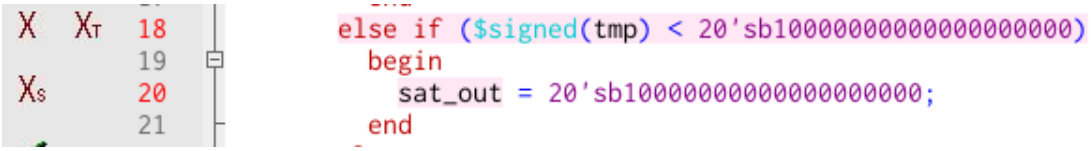
Seuraavissa esimerkeissä käydään läpi Questa Sim simulaattorin koodinkattavuusanalyysimenetelmät tapaus kerrallaan. Jokaisen tapauksen kohdalla katsotaan, että mikä on aiheuttanut puutetta kattavuuteen ja pyritään antamaan ratkaisu, jos sellaiselle on tarvetta.

3.2.1 Lausekattavuus

Kuvan 6 esimerkki lausekattavuudelle on saatu suodattimen saturaatiolohkosta. Siellä lausekattavuusvajetta aiheuttaa kuvan 7 else if -lause. Kyseisessä lohkossa tmp muuttuja ei koskaan alita vaadittua etumerkillistä 20 bittistä arvoa, joten testiohjelma ei pääse testien aikana riville 20 missä tmp arvo saturoitaisiin. Jotta voitaisiin parantaa kattavuutta, niin testiohjelmaan pitäisi koodata saturoitumiseen johtava tilanne.



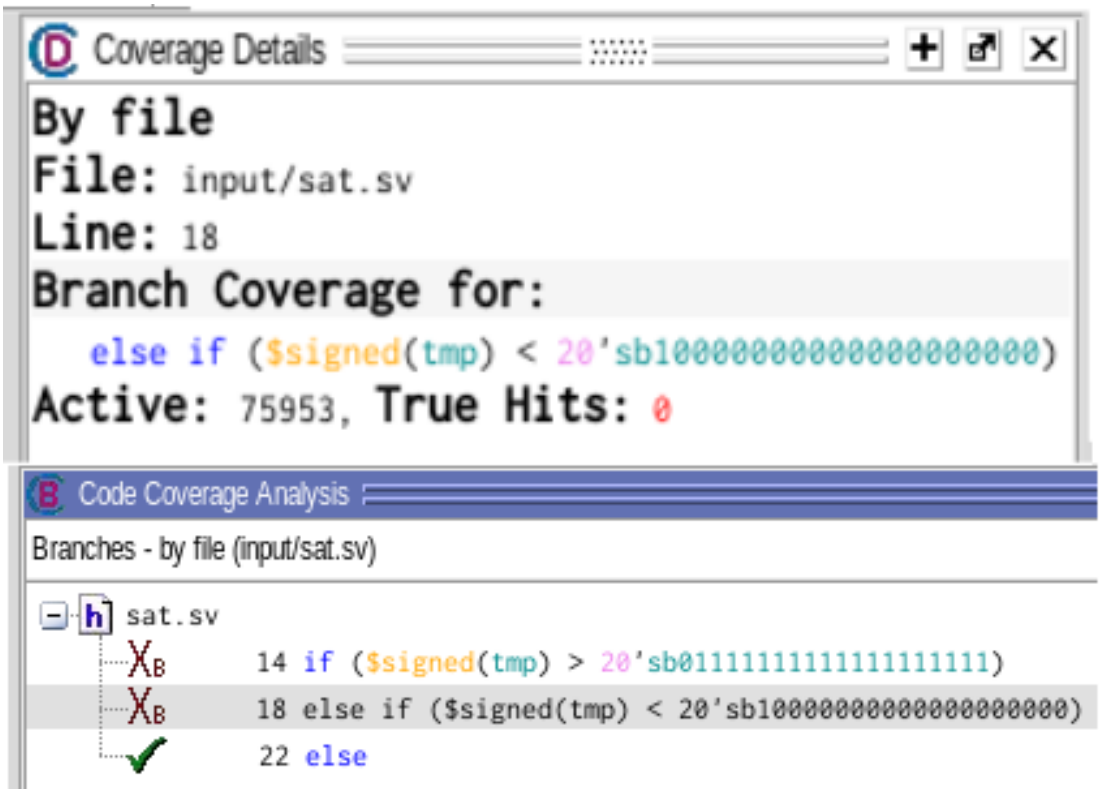
Kuva 6: Simulaattorin esitysmalli lausekattavuudelle



Kuva 7: Kattavuusvajeen aiheuttanut koodin pätkä.

3.2.2 Haarautumiskattavuus

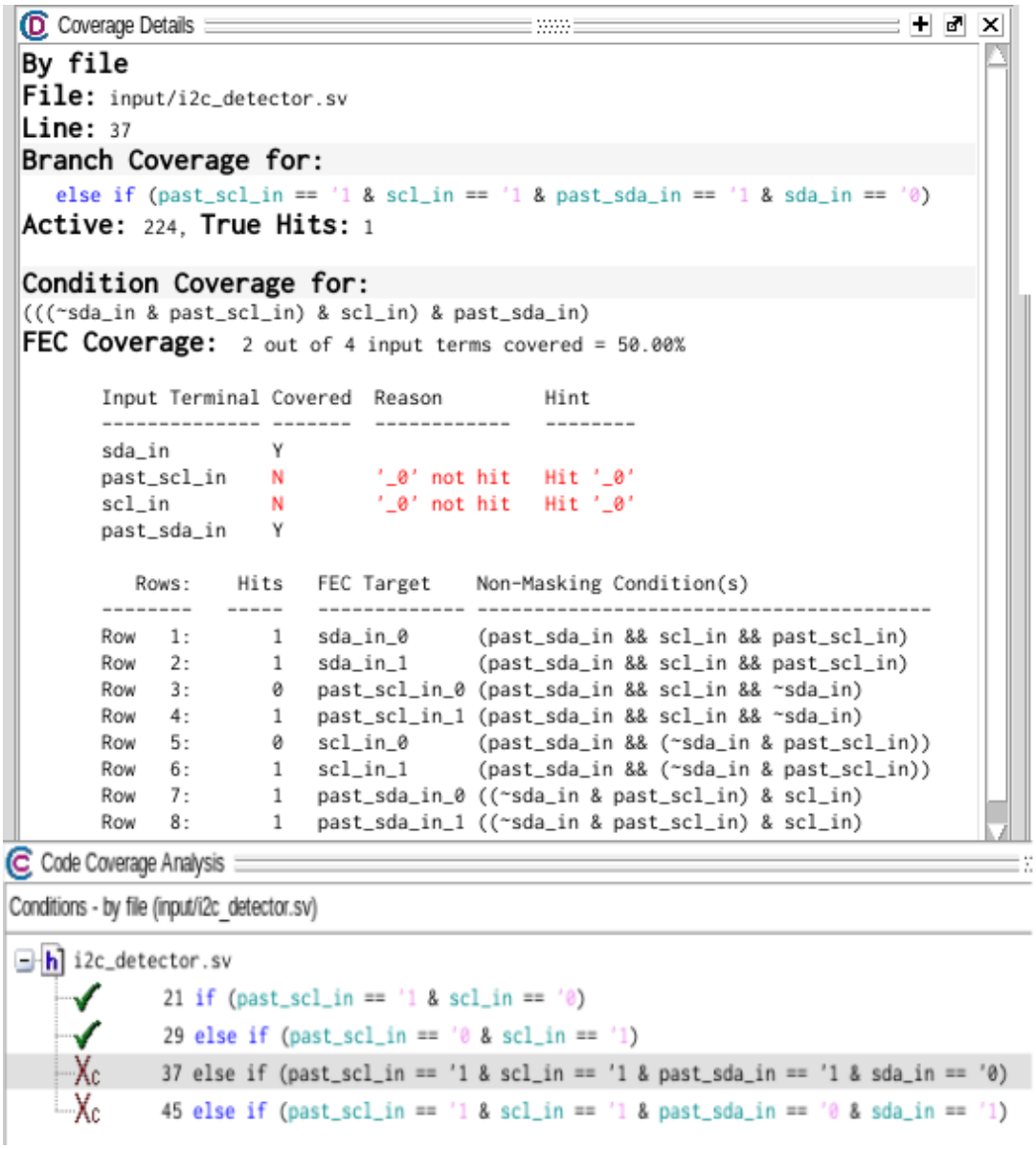
Haarautumiskattavuuden esimerkki on samasta tilasta kuin lausekattavuudella, mutta nyt ohjelma keskittyykin pelkästään vain else if-lauseeseen. Kuvasta 8 nähdään miten Questa Sim esittää haarautumiskattavuuden. Siitä voidaan huomata, että tarkasteltavassa else if-lauseessa on käyty 75953 kertaa, mutta se ei ole kertaakaan ollut tosi eli kyseisessä kohdassa on kattavuusvajetta. Kattavuutta voidaan parantaa samalla tavalla kuin edellisessä esimerkissä. Edellisen kohdan kuvasta 7 voidaan nähdä myös hyvin, miten ohjelma ilmoittaa punaisella x:llä, mitä eri kattavuusvajeita on kyseisillä riveillä. Esimerkiksi Xs merkintä tarkoittaa, että rivillä 20 on lausekattavuusvajetta.



Kuva 8: Simulaattorin esitysmalli haarautumiskattavuudelle

3.2.3 Ehtolausekattavuus

Kuvan 9 esimerkki saatiin 12C-väylän ilmaisin lohkosta. Lohkon tarkoituksena on tunnistaa I2C väylältä tulevat signaalit ja päätellä mitä sieltä lähetetään. Kun kuvasta tarkastellaan condition-saraketta, niin huomataan signaaleilla past_scl_in ja scl_in puutetta kattavuudessa. Kattavuusvaje johtuu siitä, että kaikkia FEC:in asettamia kombinaatioita ei olla käyty läpi, kun kyseiset signaalit ovat 0. Selitys tälle on kuitenkin se, että kyseiset kombinaatiot eivät ole edes mahdollisia. Kuvasta 9 näkee kyseisen else if-lausetta edeltävät ehtolauseet, jotka pakottavat kattavuusvajetta aiheuttaneet kombinaatiot poissa if-lauseen rakenteesta.

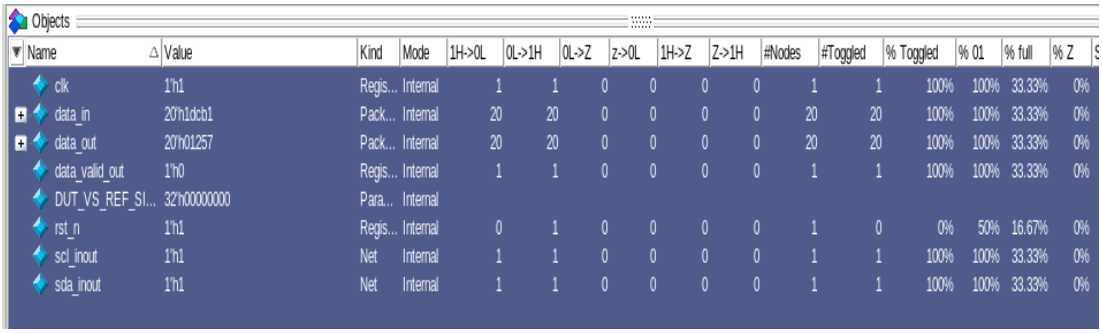


Kuva 9: Simulaattorin esitysmalli ehtolausekattavuudelle

3.2.4 Tilanvaihtokattavuus

Tilanvaihtokattavuuden analysoimiseen Questa Sim tarjoaa pari esitystapaa. Kuvan 10 esimerkki on saatu koko suodattimen testistä. Kuvasta huomataan, että kaikki paitsi rst_n signaali on päässyt läpi kattavuusanalyysistä. Tuloksista nähdään, että signaali on noussut arvoon 1, mutta ei enää takaisin arvoon 0. Kyseessä on nollaus signaali, joka vaihtaessa tilan 0:aan resetoi piirin. Piirin toimivuuden varmistamisen kannalta olisi hyvä testailla reset signaalin käyttöä testiohjelmissa. Kuva 11 on Questa Sim:n objektit -ikkunasta. Sieltä voidaan myös todeta rst_n signaalin aiheuttama kattavuusvaje. Samasta kuvasta näkee myös muille signaaleille tilanvaihtokattavuudet.

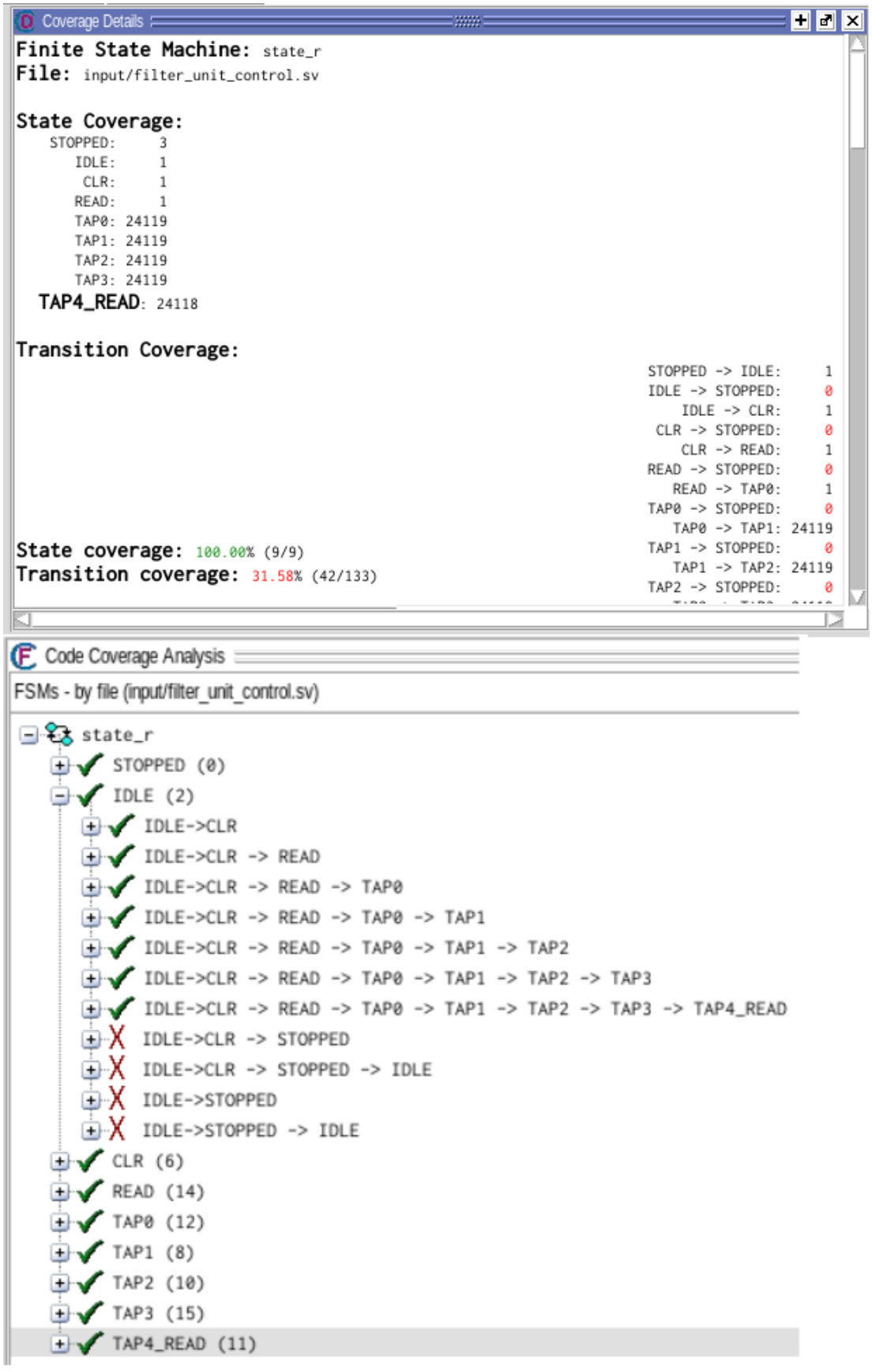
Kuva 10: Tilanvaihtokattavuuden esitysmalli G0uesta Sim:ssä



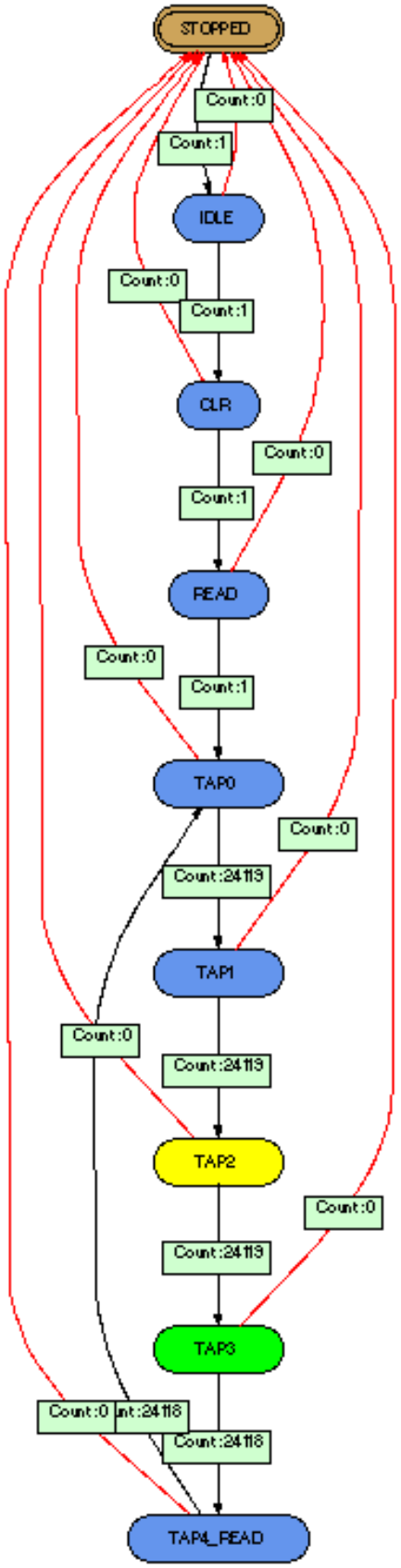
Kuva 11: Objekti ikkuna Questa Sim:stä

3.2.5 Tilakonekattavuus

Questa Sim tarjoaa myös hyvät välineet tilakoneiden analysoimiseen. Tässä esimerkissä katsotaan simulaattorin tekemää tilakonekattavuusanalyysiä suodattimen kontrollilohkon tilakoneelle. Kuvasta 12 nähdään, missä tiloissa testin aikana on käyty ja minkälaisia tilanvaihdoksia on tapahtunut. Tuloksista huomataan, että on saatu aikaiseksi 100% tilojen kattavuus, mutta vain 31.58% kattavuus tilojen muutoksille. Kun tarkastelee kuvassa olevaa IDLE tilaa, niin huomataan, että tila ei vaihdu koskaan STOPPED -tilaan. Tämä johtuu edellisessäkin kappaleessa ongelmana olleesta reset-signaalista. Tilakone on koodattu sillä tavalla, että kun reset-signaali menee nollaan, niin siirrytään STOPPED -tilaan. Sama asia voidaan myös huomata kuvan 13 tilakoneesta, joka on saatu suoraan Questa Sim:stä. Siinä punaiset viivat kuvaavat reset-signaalin reittiä ja numerot tilojen välillä kuvaavat sitä, kuinka monesti kyseinen tilanvaihdos on tapahtunut. Ratkaisu tähän olisi se, että testiin pitäisi koodata nollautumiseen johtava tilanne edes parille eri tilalle.



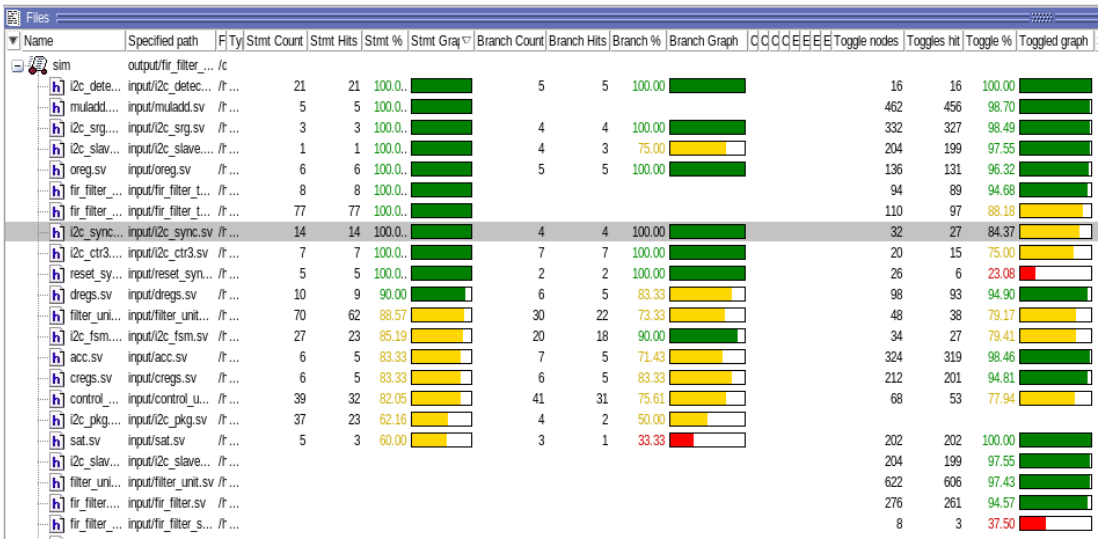
Kuva 12: Esimerkki tilakonekattavuudelle



Kuva 13: Simulaattorin esitysmalli tilakoneelle

3.2.5 Yleiskuva kattavuudesta

Ohjelman avulla voi myös saada nopeasti hyvän yleiskuvan kattavuudesta. Kuvasta 14 nähdään koodin eri lohkoille lause-, haarautumis- ja tilanvaihtokattavuudet väripalkkien ja prosenttien avulla. Mitä täydempi palkki niin sitä parempi kattavuus on saatu aikaiseksi. Kuvasta 15s saadaan tietoa jokaiselle lohkolle yksinkertaisessa raporttimuodossa. Se kertoo, että mitä eri kattavuusanalyysejä eri lohkoille on tehty ja miten hyvän kattavuuden ne ovat saaneet. Tämä on hyvä tapa kartoittaa testiohjelmasta löytyviä puutteita ja näin ollen nopeuttaa suunnittelu työtä.



Kuva 14: Kattavuusanalyysi tulokset visuaalisessa muodossa

```
================================================================================
=== Instance: /fir_filter_tb/DUT_INSTANCE/i2c_slave_1/i2c_sync_1
=== Design Unit: work.i2c_sync
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         4         4         0   100.00%
    Statements                      14        14         0   100.00%
    Toggles                         32        27         5    84.37%

================================================================================
=== Instance: /fir_filter_tb/DUT_INSTANCE/i2c_slave_1/i2c_detector_1
=== Design Unit: work.i2c_detector
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         5         5         0   100.00%
    Conditions                      12         8         4    66.66%
    Statements                      21        21         0   100.00%
    Toggles                         16        16         0   100.00%

================================================================================
=== Instance: /fir_filter_tb/DUT_INSTANCE/i2c_slave_1/i2c_ctr3_1
=== Design Unit: work.i2c_ctr3
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         7         7         0   100.00%
    Conditions                       1         1         0   100.00%
    Statements                       7         7         0   100.00%
    Toggles                         20        15         5    75.00%

--------------------------------------------------------------------------------
```

Kuva 15: Kattavuusanalyysi yksinkertaisessa raportti muodossa

4. POHDINTA

Tässä työssä pohjimmaisena ideana oli perehtyä koodinkattavuusanalyysi metodeihin ja antaa käytännön esimerkkejä simulaattorin avulla. Asiat olivat sen verran yksinkertaisia, että jos ymmärsi edes jotenkin teoriaa, niin simulaattorin tulosten analysointi oli helpohkoa. Asiaa helpotti myös Questa Sim:n kattava ja hyvä käyttöohje, josta sai kaiken tarvittavan tiedon simuloimiseen ja tulosten analysoimiseen. Ainut vaikea simulointiin liittyvä seikka oli oikeastaan vain hyvien esimerkkien löytäminen.

Parhaimmissa simulaattoriohjelmissa kuten tässäkin käytetyssä Questa Sim:ssä on myös mahdollisuus sulkea pois kattavuusanalyysit esimerkiksi tietyiltä lohkoilta, tiedostoilta ja koodinpätkiltä. Sulkemalla pois tilanteita missä ei tarvitse kattavuusanalyysejä saadaan paljon realistisempi kattavuus aikaiseksi. Esimerkiksi kappaleen 3.2.3 tilanteessa olisi voitu poissulkea FEC analyysissä turhaa kattavuusvajetta aiheuttaneet tilanteet ja näin ollen olisi saatu paljon realistisempi kuva kattavuudesta.

Koodinkattavuusanalyysit on myös paljon järkevämpi tehdä aluksi jokaiselle lohkolle erikseen. Lohkojen analysointi erikseen helpottaa huomattavasti testiohjelmien suunnittelua ja testaamista ylipäätänsä. Simulointiosiossa testiohjelma oli tehty koko suunnitelmalle, joten esimerkiksi kohdan 3.2.5 reset-ongelman korjaamien olisi ollut vaikeaa. Jos kyseistä lohkoa olisi kuitenkin testattu erikseen muista, niin testiohjelman kirjoittaminen reset-signaalille olisi ollut paljon helpompaa.

5. YHTEENVETO

Teoriaosiossa käytiin läpi viisi eri koodinkattavuusanalyysi metodia. Näistä eri koodinkattavuusanalyysi tavoista kerrottiin enemmän koodi esimerkkien avulla ja samalla käytiin läpi niiden hyödyntämisen keinoja RTL-verifioinnissa.

Simulaatio osion ideana oli simuloida Questa Sim -simulaattorin avulla SV:lla koodattua FIR-suodatinta, jolle oli luotu oma testiohjelma sen testaamista varten. Testituloksista saatiin esimerkkejä eri koodinkattavuusanalyysi metodeille. Näistä esimerkeistä etsittiin syyt kattavuusvajeelle ja tarpeen mukaan kerrottiin tapa millä päästäisiin kyseisen kohdan kattavuusvajeesta eroon.

6. LÄHTEET

[1] Tiikkainen M. (2017). Automated functional coverage driven verification with Universal Verification Methodology (Master's thesis, University of Oulu). Haettu osoitteesta http://urn.fi/URN:NBN:fi:oulu-201711033027

[2] Simpson, P. A. (2015). FPGA design: Best practices for team-based reuse (Second edition.) Cham: Springer.

[3] Barrera, B. (1998). Code coverage analysis -- essential to a safe design. Electronic Engineering, *70*(862), 41. Haettu osoitteesta https://search.proquest.com/docview/203765593?accountid=13031

[4] Questa Sim User's Manual Software Version 2019.3

[5] Raysalemi (2011) What the heck is FEC? Focused expression coverage explained. Haettu osoitteesta https://www.eejournal.com/article/20110927-fec/